UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/827,974	04/06/2001	Jason Souloglou		7224

36183 7590 09/21/2005

PAUL, HASTINGS, JANOFSKY & WALKER LLP
P.O. BOX 919092
SAN DIEGO, CA 92191-9092

EXAMINER	
YIGDALL, MICHAEL J	
ART UNIT	**PAPER NUMBER**
2192	

DATE MAILED: 09/21/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

87

Office Action Summary	Application No. 09/827,974	Applicant(s) SOULOGLOU ET AL.
	Examiner Michael J. Yigdall	Art Unit 2192

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *05 July 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on May 13, 2005 has been entered. Claims 1-30 are now pending.

Response to Arguments

2. Applicant's arguments with respect to claims 1-26 have been fully considered but they are not persuasive, as set forth in the advisory action mailed on June 14, 2005.

In response to Applicant's argument that the double patenting rejections are premature (Applicant's remarks, page 9), it should be noted that the outstanding rejections are provisional precisely because the copending applications have not yet issued as patents. As set forth in the final Office action mailed on November 15, 2004, "[these are] provisional obviousness-type double patenting rejection[s] because the conflicting claims have not in fact been patented." Accordingly, the double patenting rejections stand.

Regarding claim 13, Applicant contends that Walters fails to teach or suggest generating additional target code corresponding to the subsequent conditions when subsequently entering the same portion of subject code for which target code has already been generated upon initial translation (Applicant's remarks, page 11, last paragraph). However, what the claim recites is "generating additional target code" only "if no such target code has previously been generated."

Walters discloses entering a block of non-native code or "program code" and generating native code or "target code" for that block if the native code has not already been generated and stored in a cache (see, for example, column 3, line 54 to column 4, line 3). Importantly, if the native code is already in the cache, then it has already been generated during an "initial translation" of the non-native code. If the native code is not in the cache, then "no such target code has previously been generated," and it is generated during that subsequent entry of the non-native code. The native code generated and stored when the block is thus "subsequently entered" is in addition to or "additional" to any native code that is already in the cache, even if that native code in the cache corresponds to other non-native blocks.

Moreover, claim 13 does not clearly recite generating multiple sets of target code for the same portion of subject code, one set for the prevailing set of conditions upon initial translation and other sets for the different subsequent conditions when the subject code is subsequently encountered and the subsequent conditions are different from the prevailing conditions, as Applicant contends (Applicant's remarks, page 12, second paragraph). As noted above, what the claim recites is "generating additional target code" only "if no such target code has previously been generated." The additional target code is not generated each and every time "whenever subsequently the same portion of program code is entered," but rather is generated only "if no such target code has previously been generated." The plain language of the claim does not suggest that the "subsequent conditions" are necessarily different than the "prevailing set of conditions," much less that the "target code required to execute said portion of program code with said subsequent conditions" is necessarily different than the "target code which is required to execute that portion of program code with a prevailing set of conditions." Likewise,

"additional target code" does not necessarily mean "a second set of target code;" it could simply mean, "more target code." Therefore, multiple sets of target code are not inevitably generated for the same portion of subject code, and even when "additional target code" is generated, it is not necessarily different than the target code generated "on an initial translation." It should be noted that the claims are given the broadest reasonable interpretation and are interpreted based on the plain language of the claims and the plain meaning of the words of the claims. Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

Applicant contends that Walters only discloses a single translation of a portion of program code, not multiple translations of the same portion of program code (Applicant's remarks, page 12, third paragraph). However, as noted above, the plain language of the claim does not unequivocally establish that multiple translations are always performed for the same portion of program code. In the sense that a "translation" is construed as a pass or an "encounter," Walters clearly discloses multiple translations because that is how and why Walters employs the cache (Applicant's remarks, page 11, last paragraph). In fact, Walters expressly discloses two different code generation procedures for use with a given non-native instruction that is sometimes an exit instruction and sometimes not an exit instruction (see, for example, column 11, lines 28-34). Therefore, if the "prevailing set of conditions" is such that the instruction is an exit, then the "initial translation" generates and stores native code using the first procedure. If the "subsequent conditions" are such that the same instruction is not an exit, then native code is generated for that instruction using the second procedure. The native code for the "subsequent conditions" is not yet in the cache since the "initial translation" generates native

code for the "prevailing set of conditions," and thus "additional target code" is generated "whenever subsequently the same portion of program code is entered."

Regarding claims 1 and 15, Applicant contends that the combination of Davidson and Walters fails to teach or suggest generating different intermediate representations for the same given portion of program code in response to respective previous prevailing and subsequent conditions (Applicant's remarks, page 15, first paragraph). Again, however, as recited in the claims, the "additional intermediate representation" is not necessarily different than the intermediate representation generated and stored "on an initial translation." As noted above, Walters discloses entering a block of non-native code and generating native code for that block if the native code has not already been generated and stored in a cache (see, for example, column 3, line 54 to column 4, line 3). Davidson discloses generating an intermediate representation of program code (see, for example, the abstract). Together, Davidson and Walters teach entering a block of non-native code and generating an intermediate representation for that block if the intermediate representation has not already been generated and stored in a cache. Although Applicant contends that Walters only discloses making a single translation (Applicant's remarks, page 16, first paragraph), as demonstrated above, Walters discloses generating native code for an instruction under a "prevailing set of conditions" (the instruction is an exit instruction), and generating additional native code for the same instruction under "subsequent conditions" (the instruction is not an exit instruction).

Applicant further contends that in Walters and Davidson, there is no teaching or suggestion to determine the subsequent conditions (Applicant's remarks, page 17, first

paragraph). However, the claims do not recite determining any of the conditions, but rather "determining whether intermediate representation has previously been generated and stored for that portion of program code for the subsequent conditions." Furthermore, there is no inherent need to determine the subsequent conditions unless the subsequent conditions are necessarily different than the "prevailing set of conditions," which they are not, at least according to the plain language of the claims. Nonetheless, Walters expressly discloses determining the condition codes generated and needed by the non-native instructions (see, for example, column 4, lines 20-28), and generating native code for only those condition codes (see, for example, column 13, lines 25-31).

In terms of the dependent claims, Applicant contends that there is no teaching or suggestion in Davidson of generating and storing special-case intermediate representation of a particular subject code instruction only for the functionality required at that iteration (Applicant's remarks, page 19, first paragraph), and that Davidson fails to teach or suggest expression tuples having a plurality of functions or effects, where only a portion of those functions are generated and stored in a special-case intermediate representation of a particular subject code instruction only for the functionality required at that iteration of the instruction (Applicant's remarks, page 19, last paragraph). However, Walters discloses generating and storing special-case native code for only the functionality required at an iteration of a given non-native instruction, as noted above.

Regarding claims 11, 12 and 16, Applicant's arguments (Applicant's remarks, page 20) are analogous to the arguments addressed above. Furthermore, Walters teaches that a block of

program code can have alternative unused entry conditions or effects or functions by disclosing that native code is not generated for the unused condition codes (see, for example, column 13, lines 36-39).

In response to Applicant's argument that there is no suggestion to combine the references (Applicant's remarks, page 21, fourth paragraph), the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In response to Applicant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning (Applicant's remarks, page 23, second paragraph), it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

As set forth in the final Office action mailed on November 15, 2004, Davidson expressly discloses that the intermediate representation provides language independence (see, for example, column 3, lines 56-65), and further that the intermediate representation is used to perform optimization (see, for example, column 3, lines 36-40). Thus, Davidson teaches that the

intermediate representation enables language-independent optimization. Walters, too, discloses optimization (see, for example, column 3, lines 1-9). It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement Walters with an intermediate representation, such as taught by Davidson, so as to provide language independence. That Walters alone performs optimization does not preclude the desirability of language independence. Indeed, one of ordinary skill in the art would have been motivated to achieve such language-independent optimization in Walters. Applicant has merely concluded that because Davidson discloses a static compiler, one of ordinary skill in the art would not be motivated to add an intermediate representation to the dynamic translator of Walters "because of the prospect that the added complexity and overhead would tremendously slow down the Walters cross-compilation system" (Applicant's remarks, page 22, first paragraph). Notwithstanding the prospect of what Applicant concludes may possibly result from a combination of Walters and Davidson, the references teach that the intermediate representation would in fact enable language-independent optimization. Furthermore, the recognition that claim 14 recites a method of "dynamically translating" comprising the step of "generating an intermediate representation" is not presented as motivation to combine Walters and Davidson. The motivation to combine Walters and Davidson is found in the references themselves, as noted above.

Double Patenting

3. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed.

Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and, *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

4. Claims 1-26 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 17-54 of copending Application No. 10/164,789. Although the conflicting claims are not identical, they are not patentably distinct from each other because both sets of claims recite analogous systems and methods for translating code and generating intermediate representations.

For example, claim 1 of the present application and claim 17 of Application No. 10/164,789 both recite, on an initial translation of a portion of program code, generating and storing only the intermediate representation that is required to execute that portion of program code for a prevailing set of conditions, and, whenever subsequently the same portion of program code is entered, determining whether an intermediate representation has previously been generated and stored for the subsequent or then-prevailing conditions, and if not, generating an

additional intermediate representation required to execute that portion of program code for those subsequent or then-prevailing conditions.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

5. Claims 1-26 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 17-54 of copending Application No. 10/165,457. Although the conflicting claims are not identical, they are not patentably distinct from each other because both sets of claims recite analogous systems and methods for translating code and generating intermediate representations.

For example, claim 1 of the present application and claim 17 of Application No. 10/165,457 both recite, on an initial translation of a portion of program code, generating and storing only the intermediate representation that is required to execute that portion of program code for a prevailing set of conditions, and, whenever subsequently the same portion of program code is entered, determining whether an intermediate representation has previously been generated and stored for the subsequent or then-prevailing conditions, and if not, generating an additional intermediate representation required to execute that portion of program code for those subsequent or then-prevailing conditions.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

7. Claims 13, 19, 24 and 28 are rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Pat. No. 5,768,593 to Walters et al. (art of record, "Walters").

With respect to claim 13 (previously amended), Walters discloses a method of generating a target code representation of program code (see, for example, the abstract), the method comprising the computer implemented steps of:

on an initial translation of a given portion of the program code, generating and storing only target code which is required to execute that portion of program code with a prevailing set of conditions (see, for example, column 7, lines 52-63, which shows generating a block of native or target code for execution on an initial translation); and

whenever subsequently the same portion of program code is entered, determining whether target code has previously been generated and stored for that portion of program code for the subsequent conditions, and if no such target code has previously been generated, generating additional target code required to execute said portion of program code with said subsequent conditions (see, for example, column 7, lines 16-23 and 52-63, which shows

determining whether a block of code has been generated and stored in a table, and if it has not, subsequently generating that code).

With respect to claim 19 (previously added), Walters further discloses the limitation wherein said target code representation is generated at run time (see, for example, column 3, lines 35-53, which shows that the system operates at run time).

With respect to claim 24 (previously added), Walters further discloses the limitation wherein said steps are performed at run time (see, for example, column 3, lines 35-53, which shows that the system operates at run time).

With respect to claim 28 (new), Walters further discloses the limitation wherein whenever subsequently the same portion of program code is entered, the method comprises the computer implemented steps of:

determining whether said subsequent conditions are different from said prevailing set of conditions (see, for example, column 11, lines 28-34, which shows determining, in one instance, whether the subsequent conditions are such that the portion of program code is not an exit when the prevailing set of conditions is such that the portion of program code is an exit), and

if different, generating and storing different target code for said same portion of program code required to execute said portion of program code with said subsequent conditions (see, for example, column 11, lines 28-34, which shows two code generation procedures for generating different target code if the conditions are different).

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 1-5, 7-12, 15-18, 20-23, 25-27, 29 and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Pat. No. 5,613,117 to Davidson et al. (art of record, "Davidson") in view of Walters.

With respect to claim 1 (original), Davidson discloses a method of generating an intermediate representation of program code (see, for example, the abstract), the method comprising the computer implemented steps of:

on an initial translation of a given portion of program code, generating and storing only intermediate representation which is required to execute that portion of program code with a prevailing set of conditions (see, for example, column 8, line 63 to column 9, line 2, which shows a first translation stage for generating an intermediate language representation of program source code, and column 11, lines 10-26, which shows generating the representation one node at a time).

Although Davidson discloses storing data structures for the intermediate representation and a symbol table (see, for example, column 27, lines 8-21), Davidson does not expressly disclose the step of:

whenever subsequently the same portion of program code is entered, determining whether intermediate representation has previously been generated and stored for that portion of program code for the subsequent conditions, and if no such intermediate representation has previously been generated, generating additional intermediate representation required to execute said portion of program code with said subsequent conditions.

However, Walters discloses a cross-compiler (see, for example, the abstract) comprising the step of determining whether a code block has previously been translated and stored in a table, and if it has not, subsequently translating that portion of the code to be executed (see, for example, column 7, lines 16-23 and 52-63). This code caching feature increases efficiency and enables previously translated code to be reused (see, for example, column 4, lines 46-67).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the code generation method of Davidson with the code caching feature taught by Walters, for the purpose of increasing efficiency and enabling the reuse of previously generated code.

With respect to claim 2 (original), the combination of Davidson and Walters further discloses the limitations wherein the conditions are entry conditions (see, for example, Davidson, column 7, line 66 to column 8, line 8, which shows identifying blocks based on entry conditions), and the method comprises the computer implemented steps of:

generating an Intermediate Representation Block (IR Block) of intermediate representation for each Basic Block of program code as it is required by the program, each IR Block representing a respective Basic Block of program code for a particular entry condition (see, for example, Davidson, column 8, line 63 to column 9, line 2, which shows generating

intermediate language blocks from program source code, and column 7, line 66 to column 8, line 8, which shows that the blocks are associated with particular entry conditions);

storing target code corresponding to each IR Block (see, for example, Davidson, column 12, lines 22-29, which shows storing target machine code based on the intermediate representation); and

when the program requires execution of a Basic Block for a given entry condition, either:

(a) if there is stored target code representing that Basic Block for that given entry condition, using said stored target code (see, for example, Walters, column 7, lines 16-23, which shows using stored code if it is found for that entry condition); or

(b) if there is no stored target code representing that Basic Block for that given entry condition, generating a further IR Block representative of that Basic Block for that given entry condition (see, for example, Walters, column 7, lines 52-63, which shows generating code if it is not found for that entry condition).

As set forth above, it would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the code generation method of Davidson with the code caching feature taught by Walters, for the purpose of increasing efficiency and enabling the reuse of previously generated code.

With respect to claim 3 (original), the combination of Davidson and Walters further discloses the limitation wherein the intermediate representation of the program code is generated dynamically as the program code is running (see, for example, Walters, column 3, lines 35-53, which shows that the code is generated at run time), the method comprising the computer implemented steps of:

at a first iteration of a particular subject code instruction having a plurality of possible effects or functions, generating and storing special-case intermediate representation representing only the specific functionality required at that iteration (see, for example, Davidson, column 7, lines 24-65, which shows generating the intermediate representation in terms of tuples that represent the functionality of an instruction, and see, for example, column 8, lines 9-17, which shows that the tuples represent the effects of an instruction); and

at each subsequent iteration of the same subject code instruction, determining whether special-case intermediate representation has been generated for the required functionality required at said subsequent iteration and generating additional special-case intermediate representation specific to that functionality if no such special-case intermediate representation has previously been generated (see, for example, Walters, column 7, lines 16-23 and 52-63, which shows determining whether code has been generated and stored in a table, and if it has not, subsequently generating that code).

As set forth above, it would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the code generation method of Davidson with the code caching feature taught by Walters, for the purpose of increasing efficiency and enabling the reuse of previously generated code.

Furthermore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to perform the code generation method of Davidson dynamically as the program code is running, as taught by Walters, for the purpose of improving execution time (see, for example, Walters, column 2, lines 17-23).

With respect to claim 4 (previously amended), the combination of Davidson and Walters further discloses the limitation wherein the said special-case intermediate representation is generated and stored and an associated test procedure is generated and stored to determine on subsequent iterations of the respective subject code instruction whether the required functionality is the same as that represented by the associated stored special-case intermediate representation (see, for example, Davidson, column 13, lines 7-44, which shows using a test procedure to determine whether an instruction has the same effect or functionality as another instruction), and, where additional special-case intermediate representation is required, an additional test procedure associated with that special-case intermediate representation is generated and stored with that additional special-case intermediate representation (see, for example, column 13, line 60 to column 14, line 17, which shows generating additional classes or procedures to determine the effects of instructions).

With respect to claim 5 (previously amended), the combination of Davidson and Walters further discloses the limitation wherein the additional special case intermediate representation for a particular subject code instruction and the additional associated test procedure is stored at least initially in subordinate relation to any existing special-case intermediate representation and associated test procedures stored to represent the same subject instruction (see, for example, Davidson, column 10, lines 40-44, which shows that the intermediate representation is stored as a graph of linked nodes, i.e. the nodes are stored in subordinate relation to one another), such that upon the second and subsequent iteration of a subject code instruction, a determination of whether or not required special-case intermediate representation has previously been generated is made by performing said test procedures in the order in which they were generated and stored

until either it is determined that special-case intermediate representation of the required functionality exists, or it is determined that no such required special-case intermediate representation exists in which case more additional intermediate representation and another associated test procedure is generated (see, for example, column 14, lines 21-52, which shows performing the test procedures in order based on the flow paths between blocks, i.e. in the order the intermediate representation would have been generated and stored, and see, for example, column 16, lines 27-46, which further shows determining the order of the blocks in conjunction with the test procedures).

With respect to claim 7 (original), the combination of Davidson and Walters further discloses translating the program code written for execution by a processor of a first type so that the program code may be executed by a processor of a second type, using the generated intermediate representation (see, for example, Walters, column 3, lines 35-53, which shows translating non-native code into native code for execution).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the intermediate representation generated by Davidson for program code translation as taught by Walters, in order to enable the non-native code to be executed by the native processor.

With respect to claim 8 (original), the combination of Davidson and Walters further discloses the limitation wherein said translation is dynamic and performed as the program code is run (see, for example, Walters, column 3, lines 35-53, which further shows that the code is translated at run time).

With respect to claim 9 (original), the combination of Davidson and Walters further discloses optimising the program code by optimising said intermediate representation (see, for example, Davidson, column 22, lines 13-33, which shows optimizing the program code using the intermediate representation).

With respect to claim 10 (original), the combination of Davidson and Walters further discloses the limitation wherein the method is used to optimise the program code written for execution by a processor of a first type so that the program code may be executed more efficiently by that processor (see, for example, Davidson, column 22, lines 13-33, which shows optimizing the program code using the intermediate representation, and Walters, column 3, lines 35-53, which shows translating non-native code into native code for execution).

With respect to claim 11 (original), Davidson discloses a method for generating an intermediate representation of program code written for running on a programmable machine (see the title and abstract), said method comprising:

(i) generating a plurality of register objects for holding variable values to be generated by the program code (see, for example, column 7, lines 24-65, which shows generating the intermediate representation in terms of tuples, and column 33, lines 41-47, which shows that the tuples may serve as register objects); and

(ii) generating a plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code (see, for example, column 7, lines 24-65, which shows generating the intermediate representation in terms of tuples that serve as expression objects);

Although Davidson discloses generating intermediate language blocks from program source code (see, for example, column 8, line 63 to column 9, line 2) and storing the associated data structures and a symbol table (see, for example, column 27, lines 8-21), Davidson does not expressly disclose the limitation wherein said intermediate representation is generated and stored for a block of program code and subsequently re-used if the same block of program code is later re-entered.

However, Walters discloses a cross-compiler (see, for example, the abstract) wherein translated code blocks are stored in a table and subsequently reused if the blocks are later reentered (see, for example, column 7, lines 16-23). By enabling previously translated code to be reused, this code-caching feature increases the efficiency of the translation (see, for example, column 4, lines 46-67).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the code generation method of Davidson with the code caching feature taught by Walters, for the purpose of increasing efficiency and enabling the reuse of previously generated code.

Davidson further discloses the limitation wherein at least one block of program code can have alternative un-used entry conditions or effects or functions and said intermediate representation is only initially generated and stored as required to execute that block of program code with a then prevailing set of conditions (see, for example, column 7, line 66 to column 8, line 8, which shows that the blocks are associated with particular entry conditions, and column 8, lines 9-17, which shows that the tuples used in the intermediate representation represent the effects or functions of an instruction).

With respect to claim 12 (original), the combination of Davidson and Walters further discloses the limitation wherein for a given block of program code, it is determined whether a previously stored intermediate representation therefor was for the same now currently prevailing set of conditions and, if not, then generating and storing additional intermediate representation as required to execute the block of program code for the new now currently prevailing set of conditions (see, for example, Walters, column 7, lines 16-23 and 52-63, which shows determining whether code has been generated and stored in a table, and if it has not, subsequently generating that code).

As set forth above, it would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the code generation method of Davidson with the code caching feature taught by Walters, for the purpose of increasing efficiency and enabling the reuse of previously generated code.

With respect to claim 15 (original), see the explanation for claim 1 set forth above. The system recited in claim 15 is analogous to the method of claim 1. Note that Davidson further discloses a system and the means for performing the recited method (see, for example, the abstract).

With respect to claim 16 (original), see the explanation for claim 11 set forth above. The system recited in claim 16 is analogous to the method of claim 11. Note that Davidson further discloses a system and the means for performing the recited method (see, for example, the abstract).

With respect to claim 17 (previously added), the combination of Davidson and Walters further discloses the limitation wherein said intermediate representation of program code is generated at run time (see, for example, Walters, column 3, lines 35-53, which shows that the system operates at run time).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to perform the method of Davidson at run time, as taught by Walters, for the purpose of improving execution time (see, for example, Walters, column 2, lines 17-23).

With respect to claim 18 (previously added), the combination of Davidson and Walters further discloses the limitation wherein said intermediate representation of program code is generated at run time (see, for example, Walters, column 3, lines 35-53, which shows that the system operates at run time).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to perform the method of Davidson at run time, as taught by Walters, for the purpose of improving execution time (see, for example, Walters, column 2, lines 17-23).

With respect to claim 20 (previously added), the combination of Davidson and Walters further discloses the limitation wherein said intermediate representation of program code is generated at run time (see, for example, Walters, column 3, lines 35-53, which shows that the system operates at run time).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to operate the system of Davidson at run time, as taught by Walters, for the purpose of improving execution time (see, for example, Walters, column 2, lines 17-23).

With respect to claim 21 (previously added), the combination of Davidson and Walters further discloses the limitation wherein said intermediate representation of program code is generated at run time (see, for example, Walters, column 3, lines 35-53, which shows that the system operates at run time).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to operate the system of Davidson at run time, as taught by Walters, for the purpose of improving execution time (see, for example, Walters, column 2, lines 17-23).

With respect to claim 22 (previously added), the combination of Davidson and Walters further discloses the limitation wherein said steps are performed at run time (see, for example, Walters, column 3, lines 35-53, which shows that the system operates at run time).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to perform the method of Davidson at run time, as taught by Walters, for the purpose of improving execution time (see, for example, Walters, column 2, lines 17-23).

With respect to claim 23 (previously added), the combination of Davidson and Walters further discloses the limitation wherein said plurality of register objects and plurality of expression objects are generated at run time (see, for example, Walters, column 3, lines 35-53, which shows that the system operates at run time).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to perform the method of Davidson at run time, as taught by Walters, for the purpose of improving execution time (see Walters, column 2, lines 17-23).

With respect to claim 25 (previously added), the combination of Davidson and Walters further discloses the limitation wherein the function of generating and storing on an initial translation, the function of determining, and the function of generating additional intermediate representation are each performed at run time (see, for example, Walters, column 3, lines 35-53, which shows that the system operates at run time).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to operate the system of Davidson at run time, as taught by Walters, for the purpose of improving execution time (see, for example, Walters, column 2, lines 17-23).

With respect to claim 26 (previously added), the combination of Davidson and Walters further discloses the limitation wherein the functions of generating a plurality of register objects, generating a plurality of expression objects, and generating and storing intermediate representation are each performed at run time (see, for example, Walters, column 3, lines 35-53, which shows that the system operates at run time).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to operate the system of Davidson at run time, as taught by Walters, for the purpose of improving execution time (see, for example, Walters, column 2, lines 17-23).

With respect to claim 27 (new), the combination of Davidson and Walters further discloses that limitation wherein whenever subsequently the same portion of program code is entered, the method comprises the computer implemented steps of:

determining whether said subsequent conditions are different from said prevailing set of conditions (see, for example, Walters, column 11, lines 28-34, which shows determining, in one

instance, whether the subsequent conditions are such that the portion of program code is not an exit when the prevailing set of conditions is such that the portion of program code is an exit), and

if different, generating and storing a different intermediate representation for said same portion of program code required to execute said portion of program code with said subsequent conditions (see, for example, Walters, column 11, lines 28-34, which shows two code generation procedures for generating different target code if the conditions are different).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate these teachings of Walters into Davidson so as to avoid decoding the portion of program code more than once (see, for example, Walters, column 11, lines 35-37).

With respect to claim 29 (new), the combination of Davidson and Walters further discloses the limitation wherein said at least one block of program code has a plurality of possible alternative entry conditions or effects or functions (see, for example, Davidson, column 7, line 66 to column 8, line 17, which shows that the blocks are associated with particular entry conditions and effects or functions),

further wherein said intermediate representation is only initially generated and stored to represent only a portion of said plurality of possible alternative entry conditions or effects or functions as required to execute that block of program code with said prevailing set of conditions (see, for example, Walters, column 13, lines 25-31, which shows that code is generated for only a portion of the possible effects).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate these teachings of Walters into Davidson so as to reduce the amount of code that is generated (see, for example, Walters, column 13, lines 22-25).

With respect to claim 30 (new), the combination of Davidson and Walters further discloses the limitation wherein said intermediate representation does not represent any un-used entry conditions or effects or functions from said plurality of possible alternative entry conditions or effects or functions (see, for example, Walters, column 13, lines 36-39, which shows that code is not generated for any unused effects).

10. Claim 6 is rejected under 35 U.S.C. 103(a) as being unpatentable over Davidson and Walters, as applied to claim 5 above, and further in view of U.S. Pat. No. 6,631,514 to Le (art of record, "Le").

With respect to claim 6 (previously amended), although the combination of Davidson and Walters discloses optimizing the intermediate representation with code motion (see, for example, Davidson, column 3, lines 53-56) and using test procedures (see, for example, column 3, lines 56-63), the combination of Davidson and Walters does not expressly disclose the limitation wherein the intermediate representation is optimised by adjusting the ordering of the test procedures such that a test procedure associated with a more frequently used special-case intermediate representation is run before a test procedure associated with a less frequently used special-case intermediate representation rather than ordering the test procedures in the order in which they are generated.

However, Le discloses a translation system wherein the program code is optimized by reordering the instructions (see, for example, the abstract), for the purpose of achieving higher performance (see, for example, column 3, lines 13-22).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the code generation method of Davidson and Walters with the reordering feature taught by Le, so that more frequently used blocks and/or test procedures may be run first, thereby achieving higher performance.

11. Claim 14 is rejected under 35 U.S.C. 103(a) as being unpatentable over Walters in view of Davidson.

With respect to claim 14 (previously amended), Walters discloses a method of dynamically translating first computer program code written for a first programmable machine into second computer program code for running on a different second programmable machine (see, for example, the abstract).

Although Walters discloses translating a block of said first computer program code into a block of said second computer program code (see, for example, column 3, lines 35-44), Walters does not expressly disclose:

(a) generating an intermediate representation of a block of said first computer program code;

(b) generating a block of said second computer program code from said intermediate representation;

However, Davidson discloses generating an intermediate representation of program source code to provide language independence (see, for example, column 3, lines 56-65).

One of ordinary skill in the art would have been motivated to add language independence to the method of Walters, such that it may operate on programs written in any programming

language. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the method of Walters with the intermediate representation features taught by Davidson, for the purpose of providing such language independence.

Walters further discloses:

(c) running said block of second computer program code on said second programmable machine (see, for example, column 3, lines 35-53, which shows that the code is translated at run time and executed on the second processor), and

(d) repeating steps (a)-(c) in real time for at least the blocks of first computer program code needed for a current emulated execution of the first computer program code on said second programmable machine (see, for example, column 3, lines 35-53, which shows that the cross-compiler is operated in real time as the program is running).

Conclusion

12. The prior art made of record and not relied upon is considered pertinent to Applicant's disclosure. U.S. Pat. No. 5,850,554 to Carver discloses a compiler tool set for efficiently generating and easily managing multiple program versions of different types. U.S. Pat. No. 5,787,285 to Lanning discloses an apparatus and method for optimizing applications for multiple operational environments or modes.

13. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

MY

Michael J. Yigdall
Examiner
Art Unit 2192

mjy

ANTONY NGUYEN-BA
PRIMARY EXAMINER